Computer Vision

Séance 1

Stephane Ayache

Qarma, LIS – M2 IAAA

Motivations

Motivations

- Grande quantité d'images, de multiples applications
  - généraliste, médicale, robotique, biométrie, surveillance, …
  - apprentissage/classification, recherche d'information

- Besoin d’analyser, classer, comparer des images
  - La comparaison s'opère à partir d'une **représentation**
  - Les pixels constituent une première représentation mais très peu robustes aux variations de luminosité, échelle, occlusion, rotation, …

Motivations





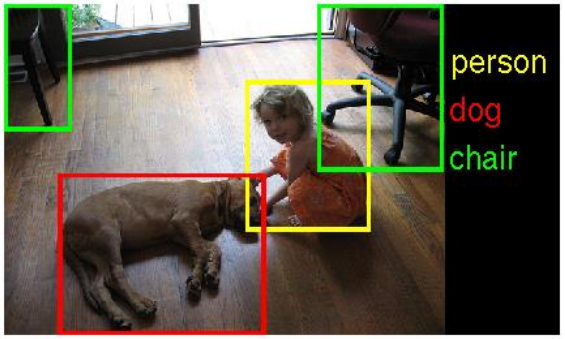




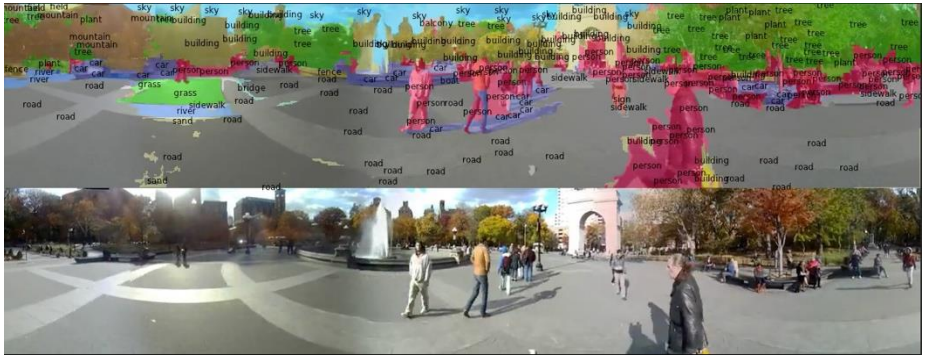


Motivations

- Interprétation de scènes, détection d'objets
  - Plusieurs niveau d'interprétations, signal vs sémantique (cf cerveau humain)
  - Problème difficile !



- Vers un très grand nombre d'images d'apprentissage et de classes ⇒ Big data !
  - Les approches classiques ne passent pas à l'échelle
  - Un apprentissage semi-supervisé et/ou incrémental peu aider..

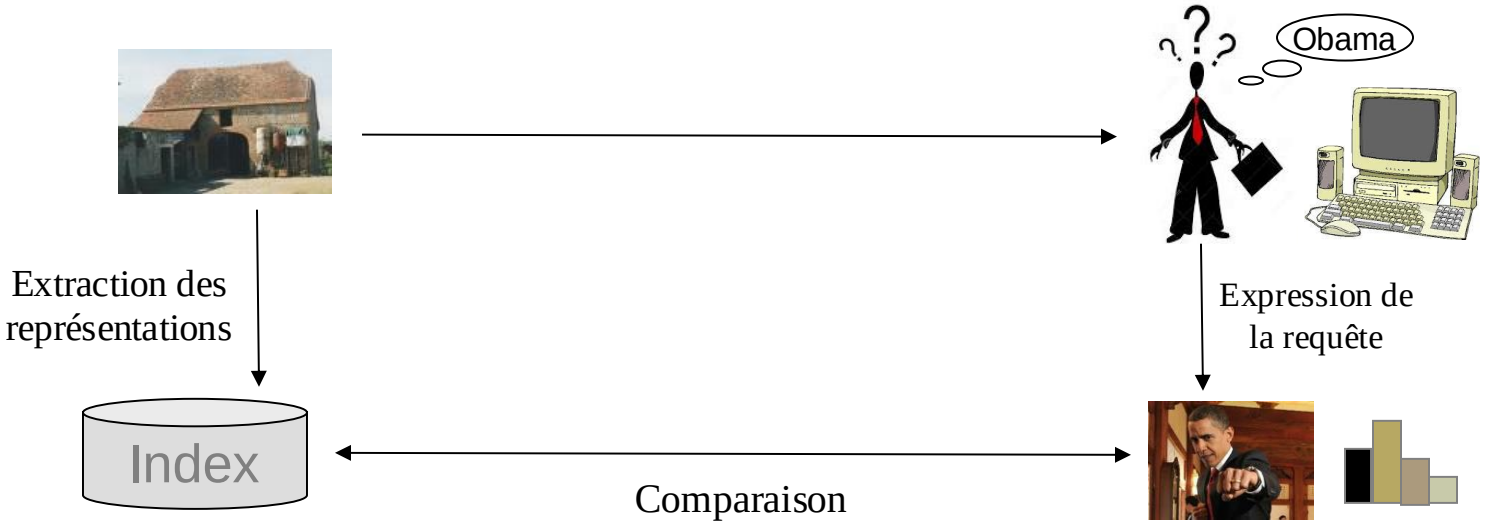


Outline

- Computer vision avant 2012
  - L'ancêtre du CBIR : histogramme de couleur...
  - Filtrage d'images, convolutions, points d'intérêts
  - Paradigme "Bag of Visual Words"
  - Pooling et pyramidal features
- Quelques mots sur l'apprentissage de dictionnaires (sparse coding) pour l'image
- Convolutional Neural Network

CBIR : Content-based Image Retrieval

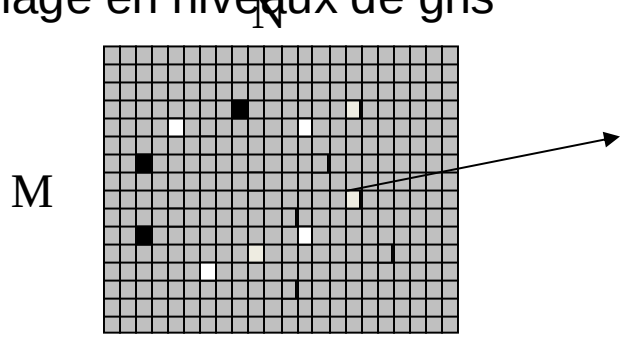
- Recherche d'images par le contenu (début 1990s)
  - Interrogation par une image exemple (ou représentation)



- Besoin d'une représentation la plus discriminante possible : descripteur d'images
  - Couleur, texture, forme, ...

Image : la structure de donnée

- Un tableau de NxM pixels
  - Une matrice 2D NxM
  - Exemple : Image en niveaux de gris

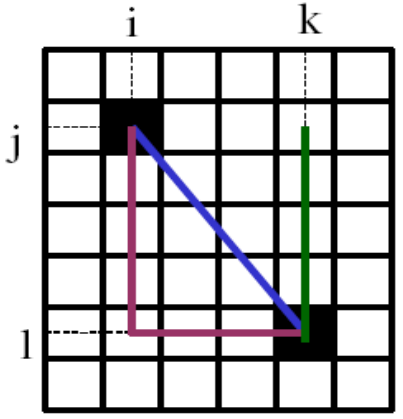


L valeurs possibles

Noir = $0 \leq l < L$ = Blanc

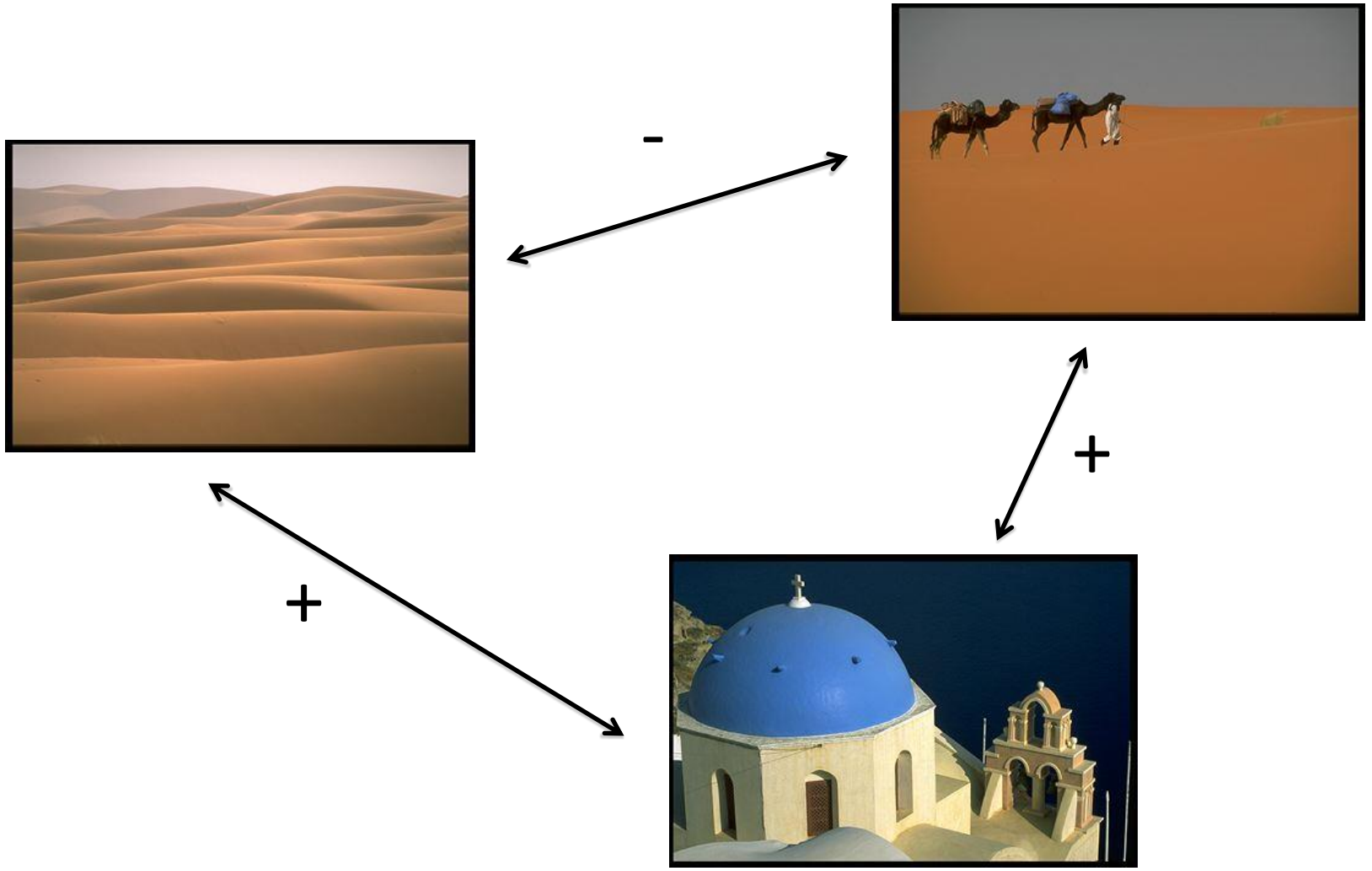
avec L=256, 1 pixel = 1 octet

- Un pixel est un élément atomique de l'image
  - Binaire, Intensité, Couleur (RGB), …
  - Dimension?
  - Distance



- Euclidean Distance
  $\sqrt{(k - i)^2 + (l - j)^2}$
- Block Distance (Manhattan)
  $| k - i | + | l - j |$
- Chess Distance
  $\max(| k - i | , | l - j |)$

Représentation et correspondance



Quelques descripteurs d'images

Quelques espaces de couleur

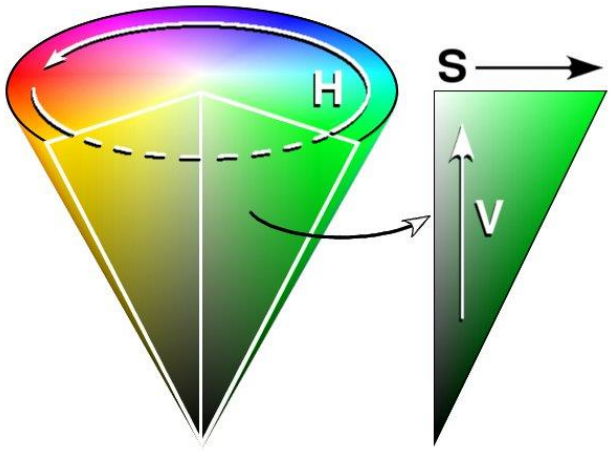
- Linéaires :
  - RGB : Rouge, vert, bleu.
    - Système de couleur additif
  - YCbCr : Luminance, Chrominances (L-rouge, L-bleu)

$$\begin{vmatrix} +0.299 & +0.587 & +0.114 \\ -0.169 & -0.441 & +0.500 \\ +0.500 & -0.418 & -0.081 \end{vmatrix} \begin{matrix} R \\ G \\ B \end{matrix} = \begin{matrix} Y \\ C_r \\ C_b \end{matrix}$$

$$\begin{vmatrix} +1.000 & +0.000 & +1.402 \\ +1.000 & -0.344 & -0.704 \\ +1.000 & +1.772 & +0.000 \end{vmatrix} \begin{matrix} Y \\ C_r \\ C_b \end{matrix} = \begin{matrix} R \\ G \\ B \end{matrix}$$

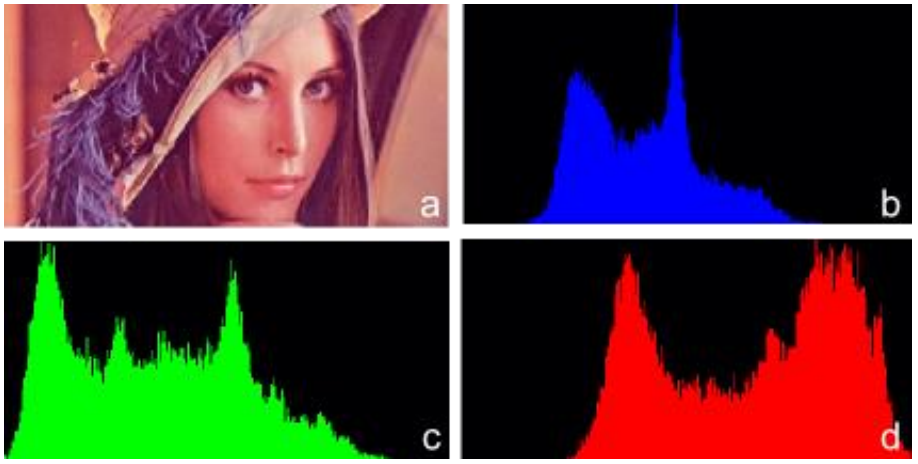
- Non linéaires :
  - HSV : Hue, Saturation, Value,
  - LAB : Luminance, Chrominances

  ⇒ Les distances euclidiennes entre couleurs sont plus proches des différences perçues par les humains



Histogrammes de couleur

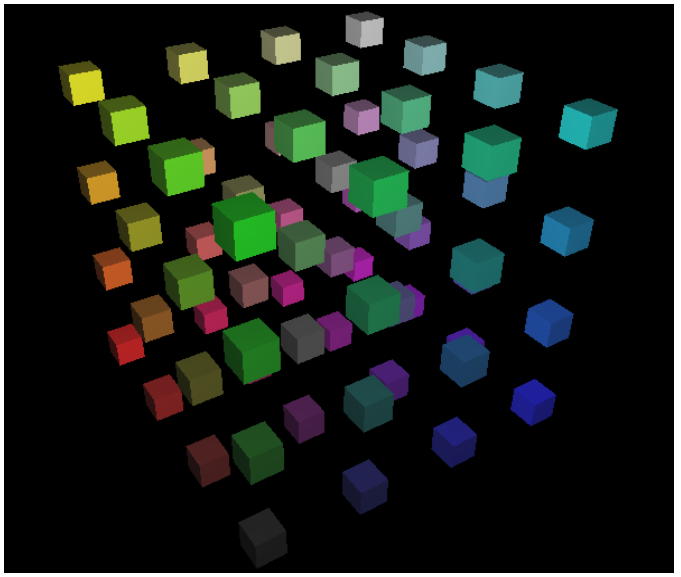
- Un histogramme par canal de couleur :
  - Histogramme composante par composante ou histogramme unique si l'image est monochrome,
  - Discrétisation : l'espace des valeurs de l'intensité est décomposé en intervalles complémentaires (batons ou « bins »),
  - Pour chaque intervalle, on calcule le pourcentage de pixels de l'image dont l'intensité s'y trouve



  - Le vecteur de ces pourcentages est le descripteur
  - Possibilité de considérer trois histogrammes pour les images en couleur

Histogrammes de couleur

- Multidimensionnels :
  - L'espace des valeurs de pixels est multidimensionnel (en général de dimension 3)
  - L'espace des valeurs est décomposé (discrétisé) en parallélépipèdes complémentaires (« bins »)
  - Mesure les coocurrences entre canaux de couleur
  - Pour chaque parallélépipède, on calcule le pourcentage de pixels de l'image dont l'intensité s'y trouve,
  - Ce tableau est le descripteur (on peut toujours le voir comme un vecteur si on sérialise les composantes).



> python color_histogram3D.py

Descripteurs à base de points d'intérêt

- Points de « haute courbure » ou « coins »,
  - Points géométriques « singuliers » de la surface Image[i][j]
- Extraits par divers **filtres** [Schmid 2000] :
  - Calcul des dérivées spatiales à une échelle donnée,
  - Convolution par des dérivées de gaussienne (Sobel, …)
  - Construction d'invariants par une combinaison appropriée de ces diverses dérivées (module du gradient par exemple),
  - Chaque point est sélectionné puis représenté par un ensemble de valeurs de ces invariants (histogramme de directions)
- L'ensemble des points sélectionnés est organisé topologiquement (relations entre points « voisins ») : SIFT
  - Descriptions très volumineuses !

Filtrage

- Le traitement d'un pixel dépend de ses voisins
- Permet de nombreuses opérations sur les images
  - Flou, points d'intérêt, contour, réhaussement, …
- Espace Fourier / Spatial
  - Une multiplication dans le domaine de Fourier équivaut à une convolution dans le domaine spatial
- Opérateur de convolution

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(x-t) \cdot g(t) \cdot dt = \int_{-\infty}^{+\infty} f(t) \cdot g(x-t) \cdot dt$$

Filtrage : convolution

- Opérateur de convolution
  - Parcourt de l'image en glissant une fenêtre : le noyau de convolution
- Approximation dans un voisinage de pixels
  - Typiquement 3x3, 5x5, 7x7, ...
- Exemple en 1D

[1 2 3 4 5 6 7]
[2 3 1] → 2 + 6 + 3 = 11

[1 2 3 4 5 6 7]
[2 3 1] → 4 + 9 + 4 = 14

[1 2 3 4 5 6 7]
[2 3 1] → 6 + 12 + 5 = 23

Et ainsi de suite... à la fin, la réponse de la convolution est le vecteur [11 14 23 29 35]

Quelques noyaux pour image

- Filtre « Moyenneur »

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Alternative plus précise

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

- Intègre plus de 80% des approximations discrètes d'une gaussienne
- Génère du « flou » ➔ diminue le contraste

Détection d'orientations

- Noyaux des gradients horizontaux et verticaux :

Opérateurs de Sobel

S_x

-1	0	1
-2	0	2
-1	0	1

S_y

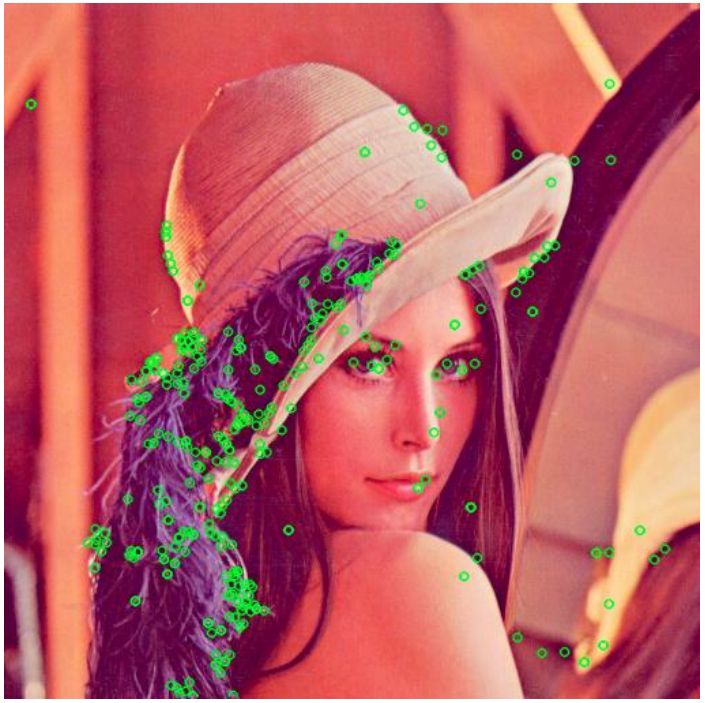
-1	-2	-1
0	0	0
1	2	1

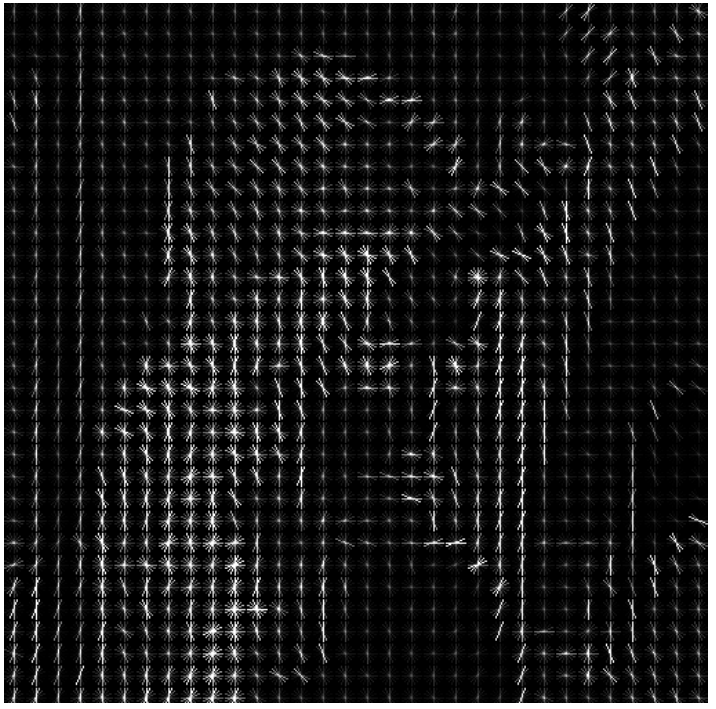
- Convolution par ces noyaux → détection des orientations

- Un descripteur possible : Histogramme des directions

Histogramme des gradients orientés (HOG) [Dalal, 2005]

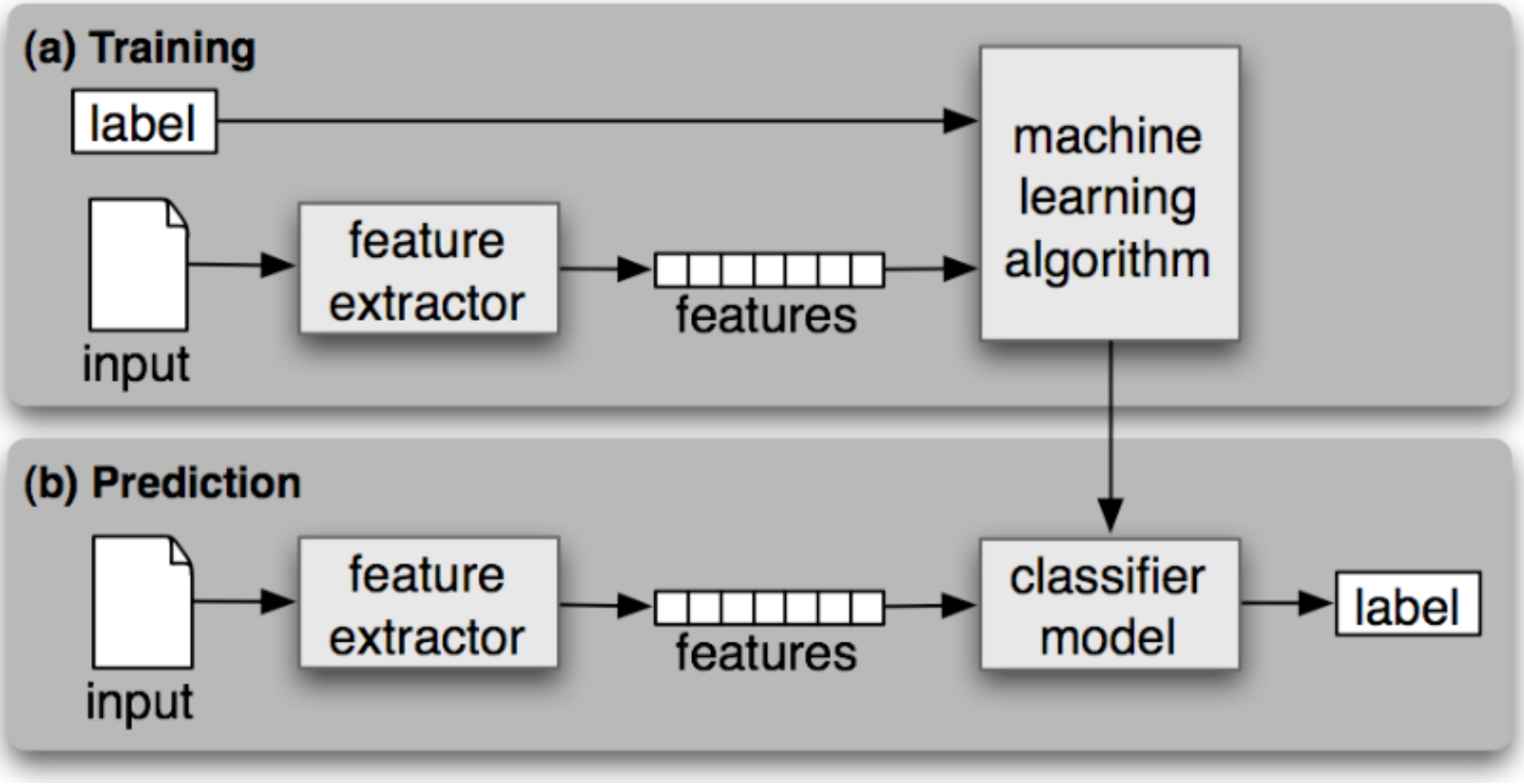
- Norme du gradient : $\mathbf{G} = \sqrt{\mathbf{G_x}^2 + \mathbf{G_y}^2}$
- Direction du gradient : $\mathbf{\Theta} = \arctan\left(\frac{\mathbf{G_y}}{\mathbf{G_x}}\right)$
- Seuil sur la norme ⇒ détection de points d'intérêts





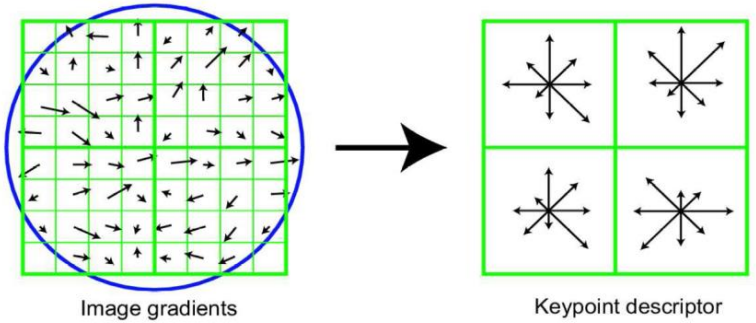
Classification d'images "classique"

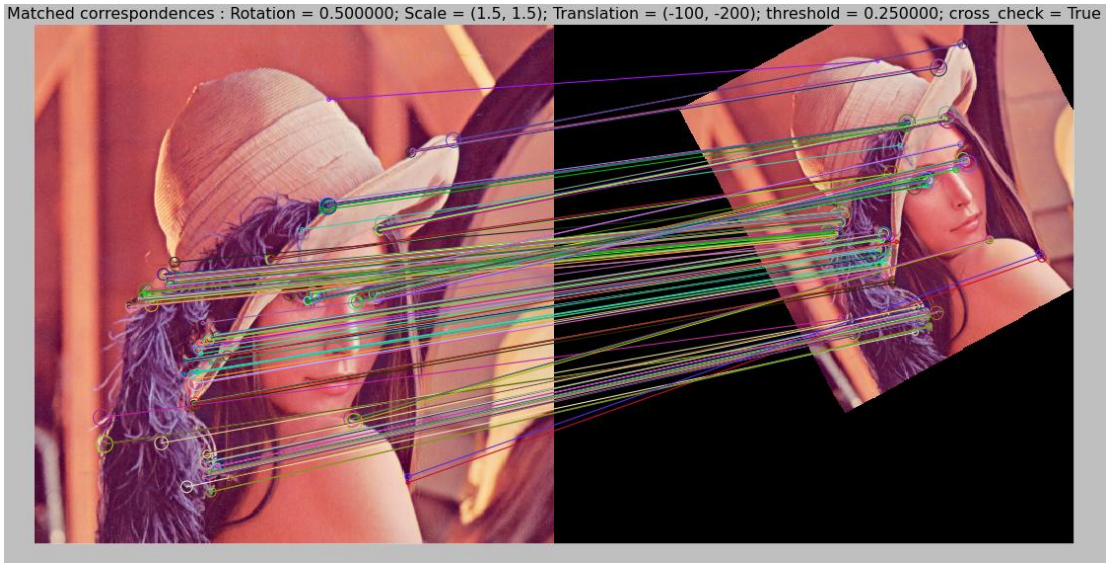
- Architectures "plates" (ou shallow)



Classification d'images par "sac de mots visuels"

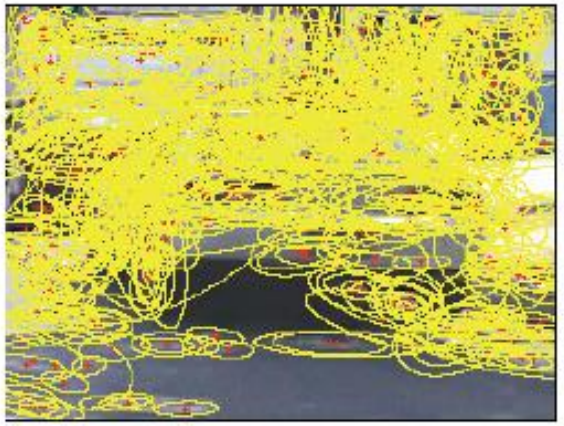
Descripteur local





- Descripteur discriminant extrait autour de points d'intérêts
- Histogramme d'orientations normalisé
- Invariant aux transformations : translation, échelle, rotation, affine
- Il en existe plusieurs : SIFT, SURF, ORB, BRIEF
- Le plus populaire est le SIFT [Lowe, 1999], mais breveté

Stratégies d'échantillonage



Sparse, at interest points

[Csurka et al. 2004]
[Fei-Fei & Perona, 2005]
[Sivic et al. 2005]

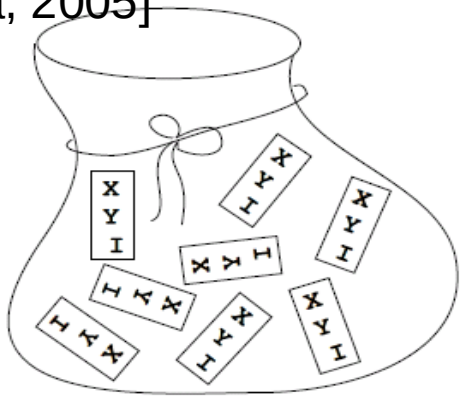
Dense, uniformly

[Vogel & Schiele, 2003]

[Fei-Fei & Perona, 2005]

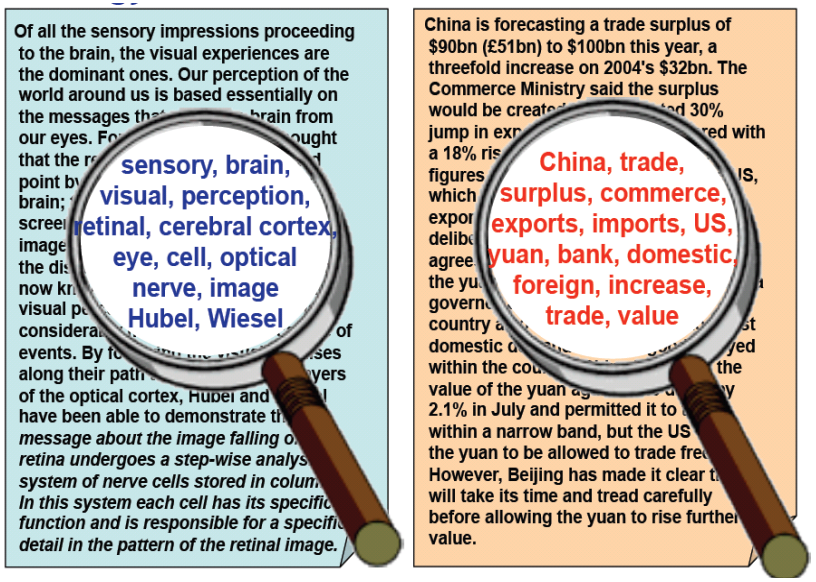
Randomly

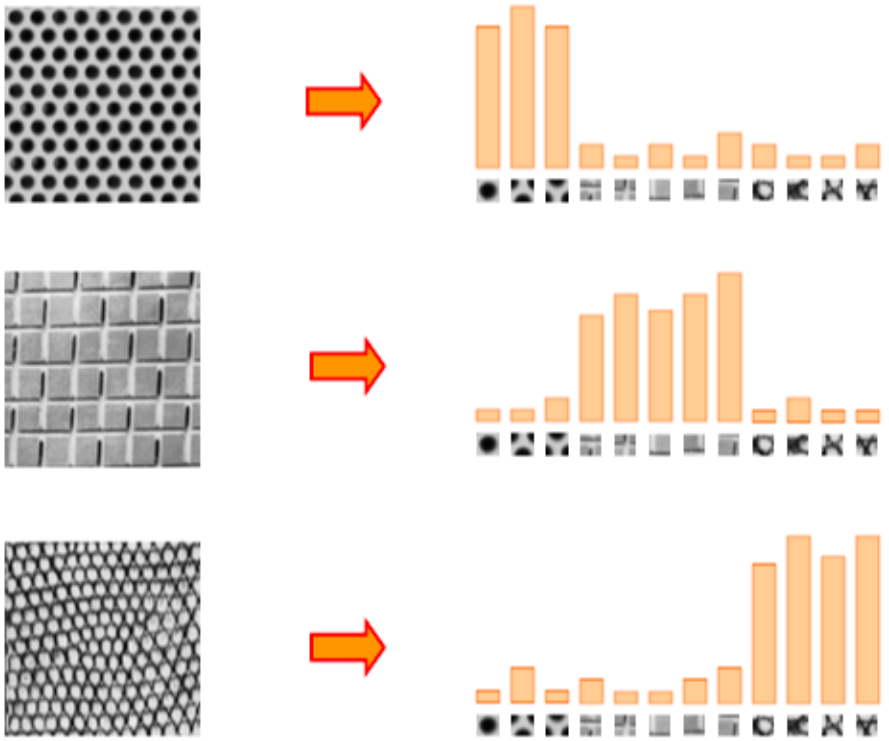
[Vidal-Naquet & Ullman, 2002]



A set of points

Représentation par "sac de mots" (BoW)

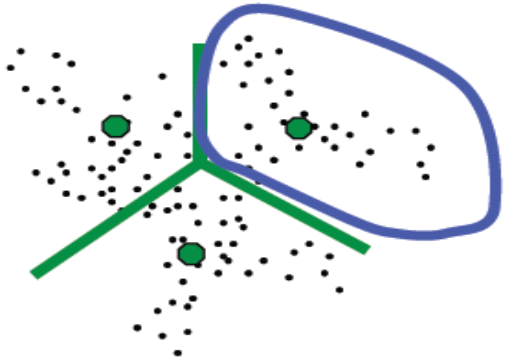




- Plusieurs inspirations :
  - Représentation de documents textuels [Salton, 1983]
  - Perception de la texture → les textons [Julesz, 1981]
- L'ordre et la caractérisation spatiale importent peu
- Nécessite un dictionnaire (mots, textons, mots visuels)

Représentation par "sac de mots" (BoW)

- Les mots visuels : regroupement (clustering) de descripteurs locaux



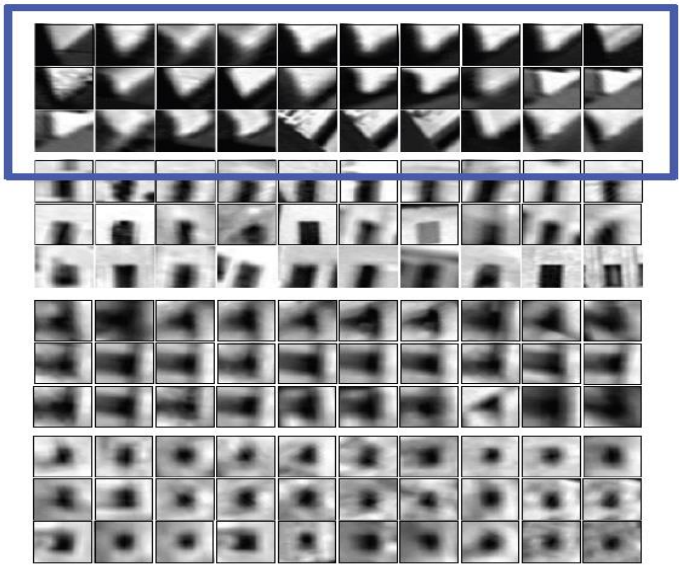
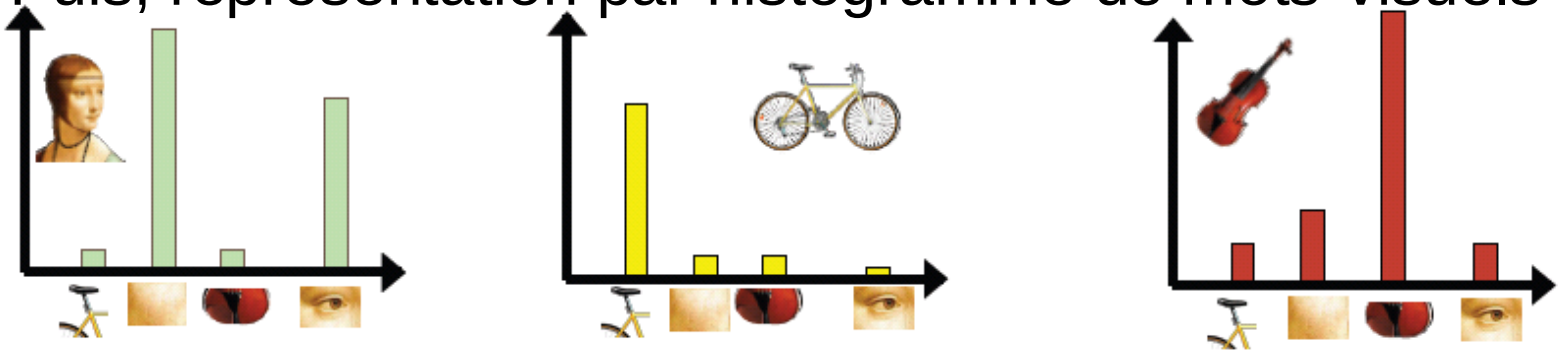
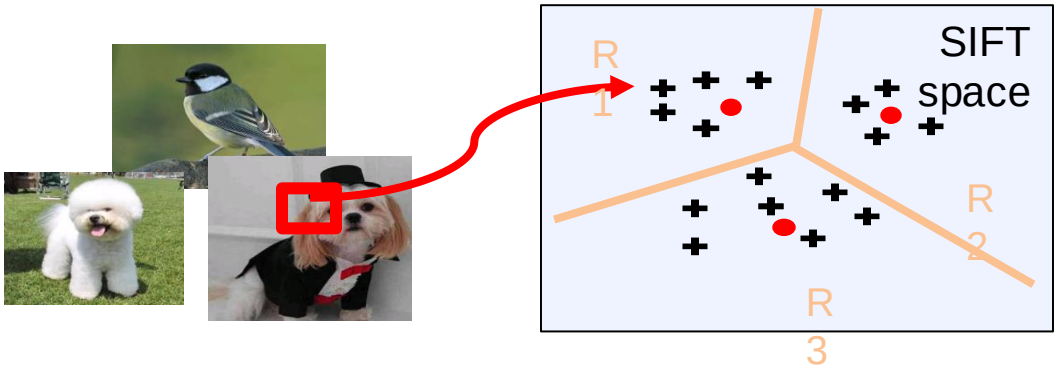


Figure from Sivic & Zisserman, ICCV 2003

- Typiquement : SIFT + Kmeans ⇒ Codebook (le dictionnaire)
  - Puis, représentation par histogramme de mots-visuels



Codebook learning & Vector Quantization



- Apprentissage du codebook (non-supervisé)

  - Kmeans $q_{ki} = \underset{k}{\operatorname{argmin}} \|\mathbf{x}_i - \mu_k\|^2$

  - GMM $p(\mathbf{x}|\mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^D \det \Sigma_k}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^\top \Sigma_k^{-1}(\mathbf{x}-\mu_k)}$

  - Supervisé : un codebook par classe, LDA, Sparse coding, ...

- Quantification des vecteurs

  → Assignation dles vecteurs (SIFT) aux clusters

  - Hard assignment (plus proche voisin) $q_{ki} = \underset{k}{\operatorname{argmin}} \|\mathbf{x}_i - \mu_k\|^2$

  - Soft assignment (selon la distance aux clusters) $q_{ki} = \frac{p(\mathbf{x}_i|\mu_k, \Sigma_k)\pi_k}{\sum_{j=1}^{K} p(\mathbf{x}_i|\mu_j, \Sigma_j)\pi_j}, \quad k = 1, \ldots, K$

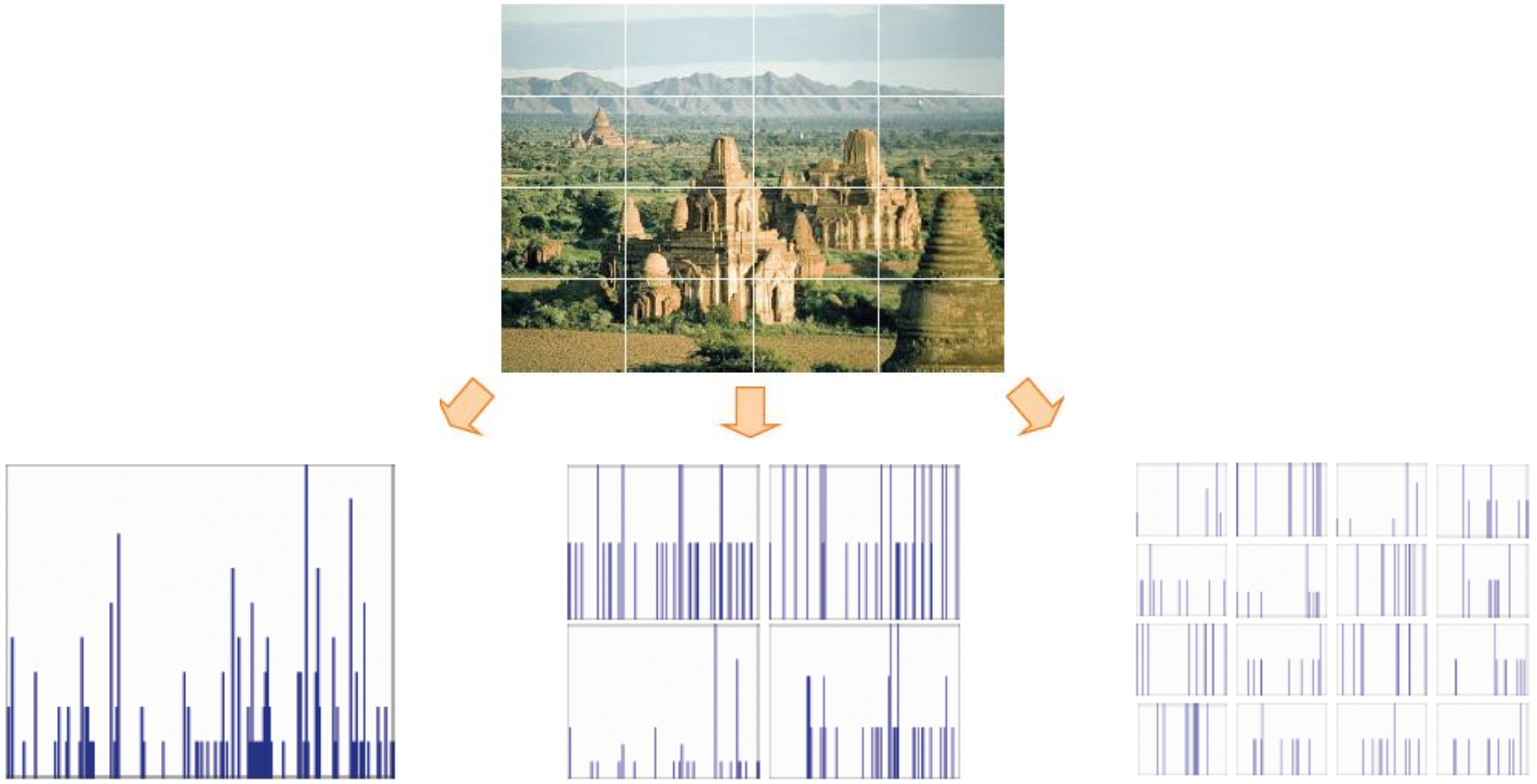
BoW : vector quantization (*vq* ou *coding*)

- Hard assignment
  - Augmente la sparsité des représentations
  - Génère de l'instabilité selon la taille du codebook
- Soft assignment
  - Représentation dense, peu redondante
- Stratégie hybride
  - ie : soft assignment sur les K plus proches voisins

Bag of Words issues

- Taille et algorithme d'apprentissage du codebook ?
  - Trop petit: mots visuels peu représentatifs de la base d'apprentissage
  - Trop gros: problème de quantification, sur-apprentissage
- Quelle stratégies d'échantillonage, quantification et pooling ?
- Passage à l'échelle ?
  - Millions d'images => milliards de SIFT.. (tirage aléatoire)
- Quelle classification à partir de cette représentation ?
- Comment intégrer la distribution spatiale des descripteurs ?
- Apprentissage joint de la représentation et du classifieur ?

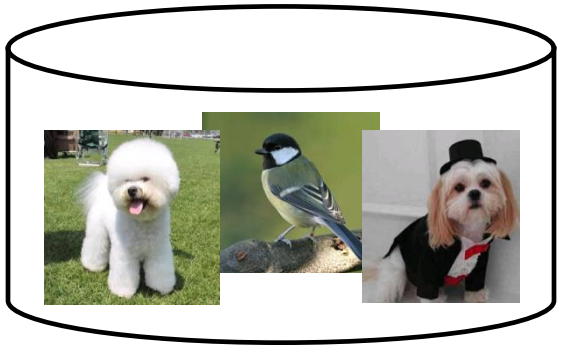
Spatial pyramid matching



➔ Pooling spatial [Lazebnik & al., 2006]

BoW : classification d'images

Codebook Learning



Dense/Sparse SIFT

K-means

codebook

Image Classification

Dense/Sparse SIFT

VQ Coding

Spatial Pyramid Pooling

Nonlinear SVM

Combining multiple descriptors

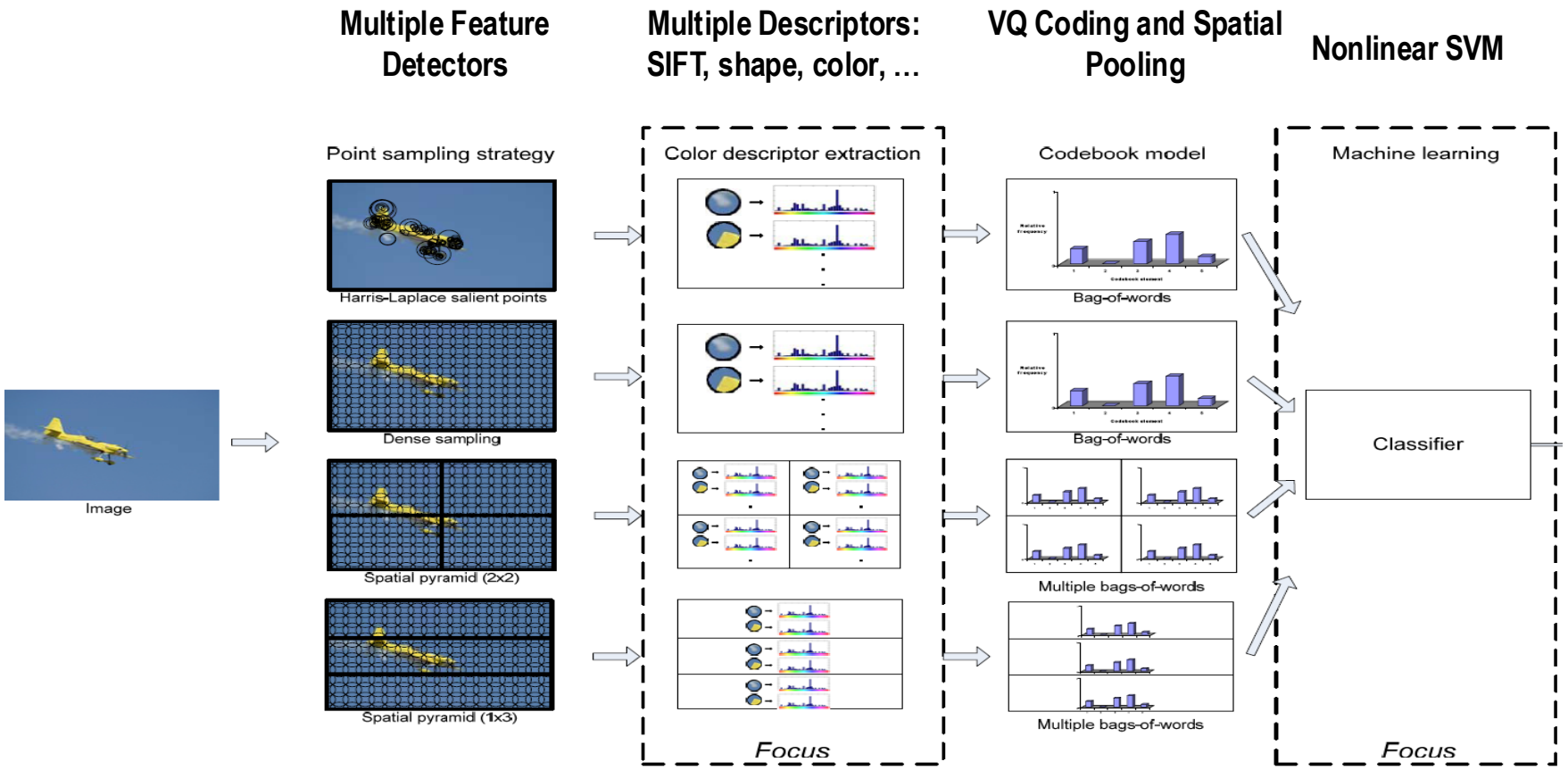


Diagram from SurreyUVA_SRKDA, winner team in PASCAL VOC 2008

Classification d'images par Sparse Coding

Sparse Coding

- Codage du signal avec contrainte de sparsité [Olshausen and Field, 1997]
  - Apprentissage de dictionnaire (codebook) pour la représentation parcimonieuse du signal

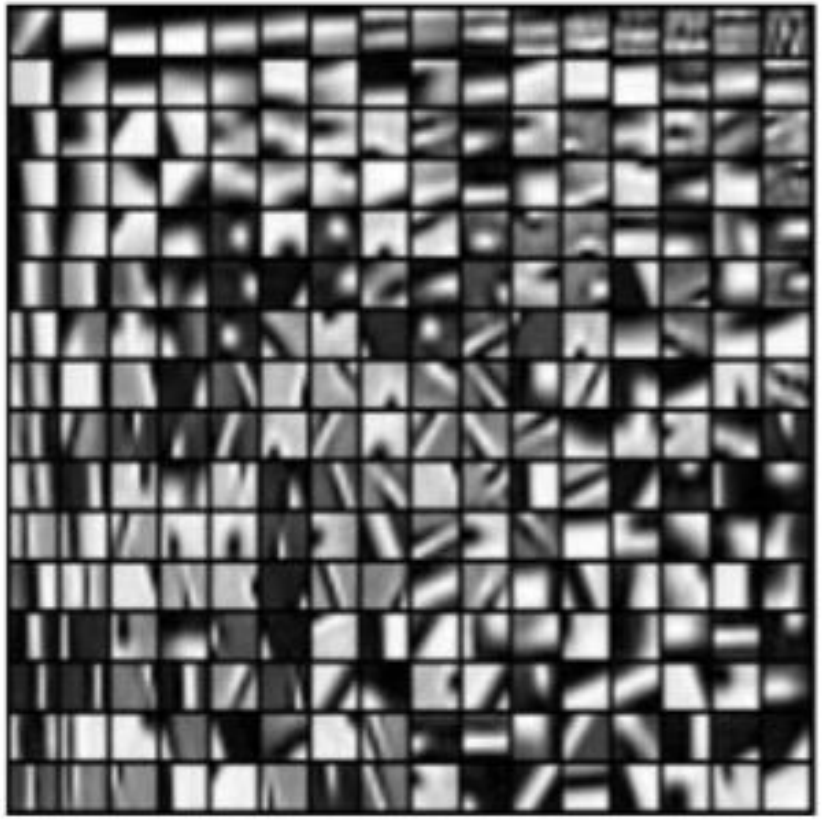
$$\min_{\boldsymbol{\alpha}_i, \mathbf{D} \in \mathcal{C}} \sum_i \underbrace{\frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2}_{\text{data fitting term}} + \underbrace{\lambda \psi(\boldsymbol{\alpha}_i)}_{\text{sparsity-inducing regularization}}$$

  - La fonction ψ induit la parcimonie sur les alpha
    - Pseudo-norme L0 : $\#\langle_i \neq 0$ (NP-hard)
    - Norme L1 : $©|\langle_i|$ (~convex)

⇒ [Olshausen and Field, 1997], [Engan et al., 1999], [Lewicki and Sejnowski, 2000], [Aharon et al., 2006] , [Roth and Black, 2005], [Lee et al., 2007]

Sparse Coding

- Nombreuses applications
  - Image denoising [Elad and Aharon, 2006]

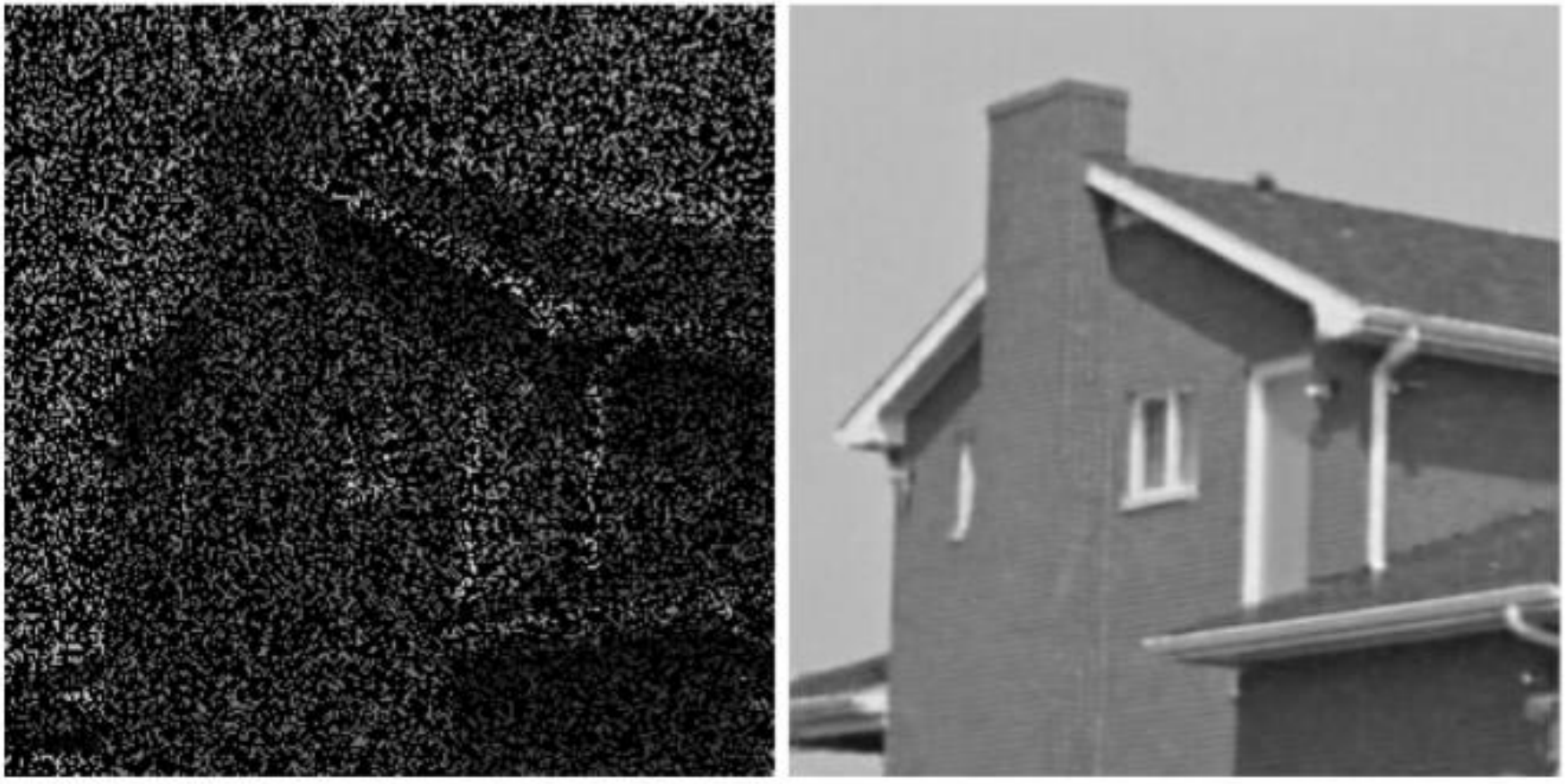


Sparse Coding

- Nombreuses applications
  - Image denoising [Elad and Aharon, 2006]

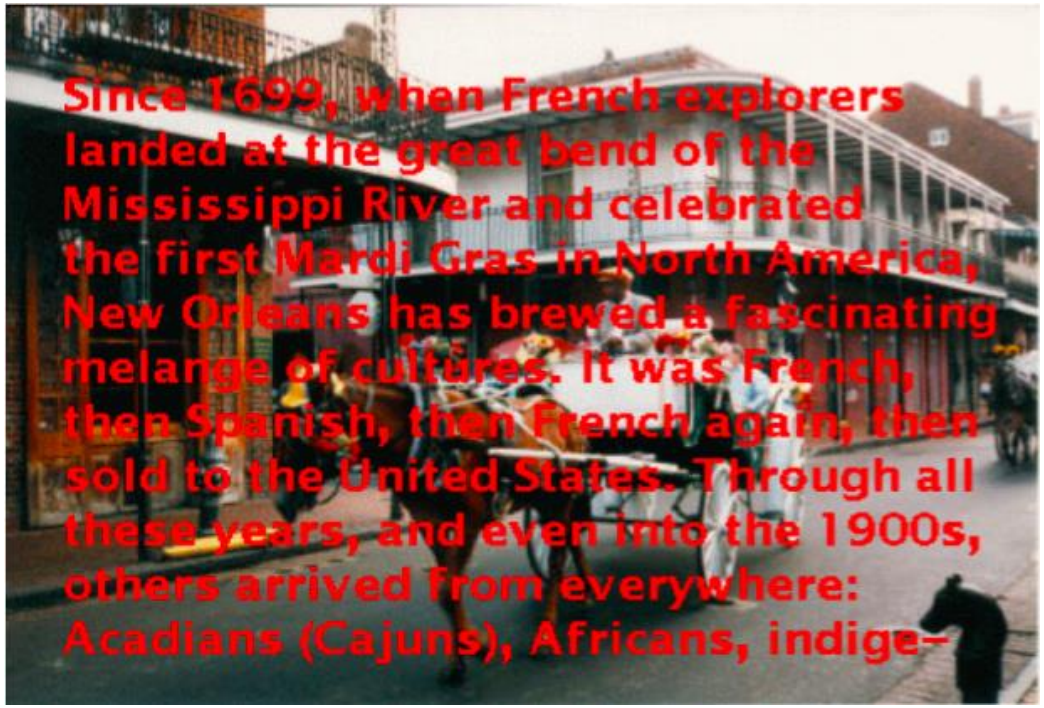
Sparse Coding

- Nombreuses applications
  - Image restoration [Mairal, Sapiro, and Elad, 2008]



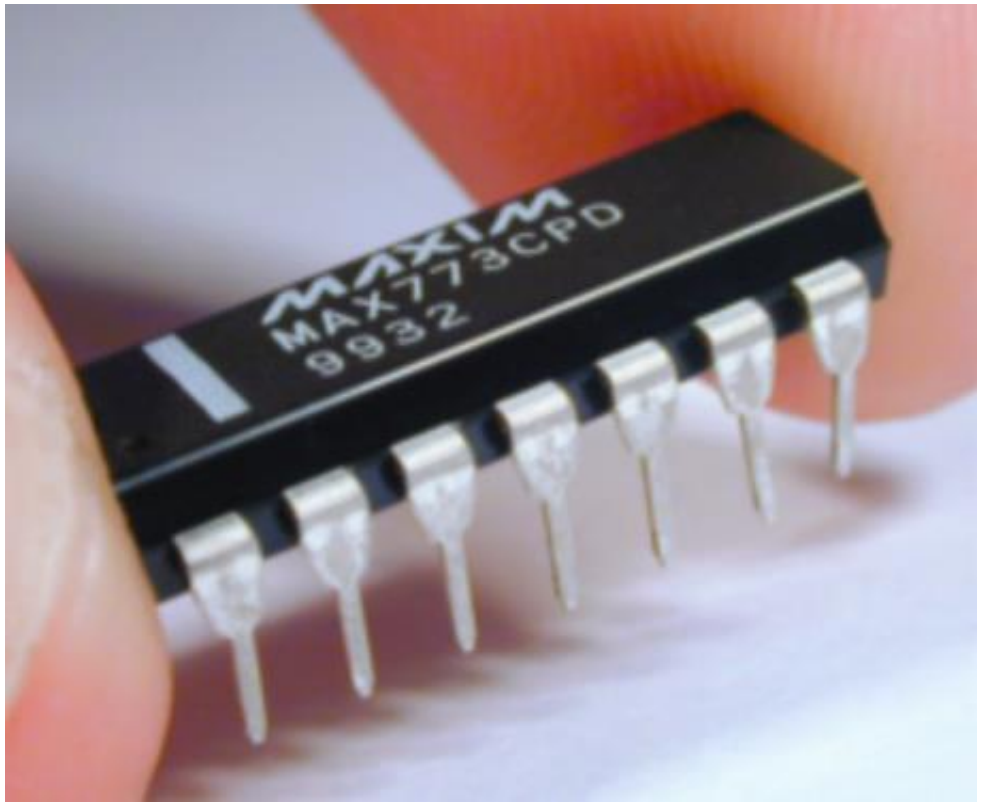
Sparse Coding

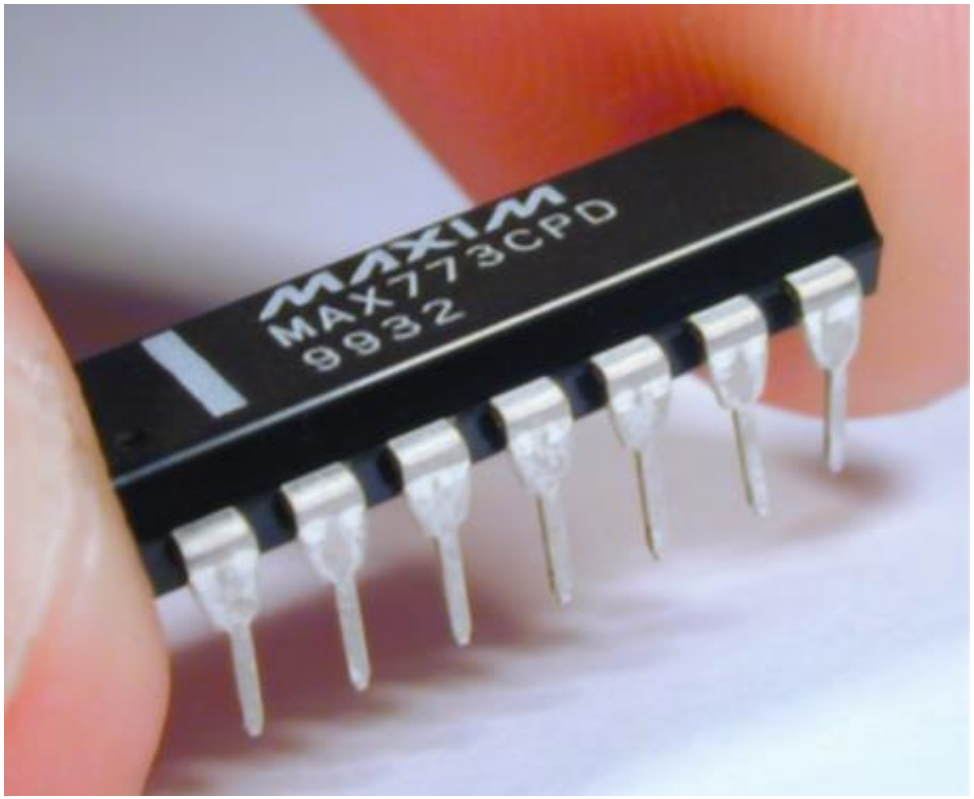
- Nombreuses applications
  - Inpainting [Mairal, Elad, and Sapiro, 2008b]



Sparse Coding

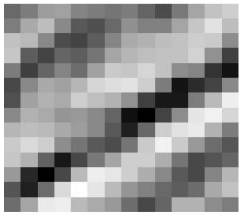
- Nombreuses applications
  - Digital zooming [Couzinie-Devy, 2010]





Sparse Coding

- Un patch de 14x14 pixels est représenté par 196 valeurs (souvent redondantes)



- Problème: Pouvons nous apprendre une meilleure représentation ?

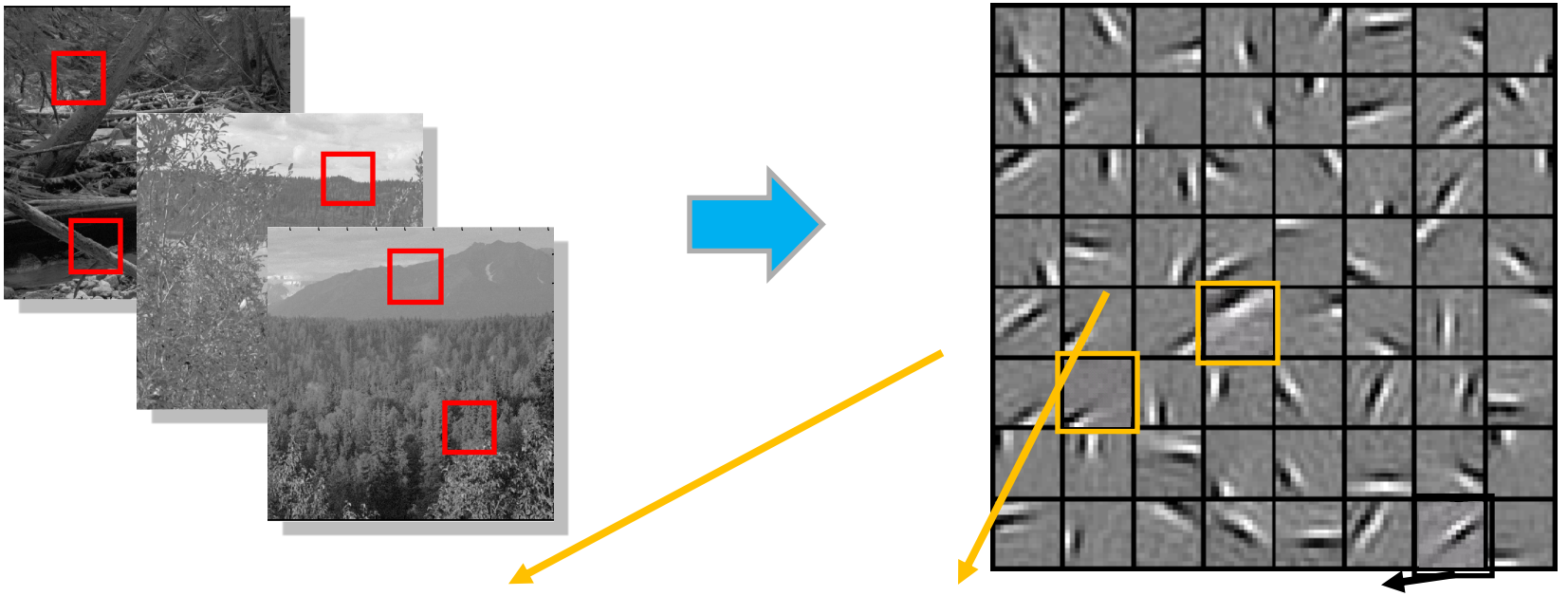
Unsupervised feature learning






Etant donné un ensemble d'images, pouvons nous apprendre une meilleure représentation que les pixels ?

Sparse coding : illustration



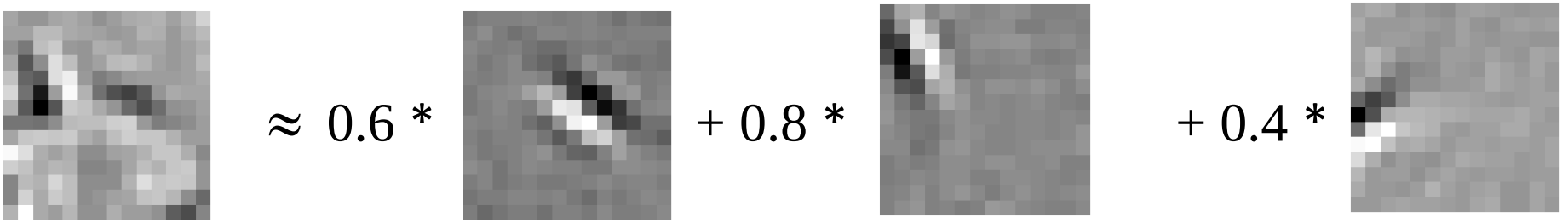
Exemple test

$$X \approx 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{63}$$

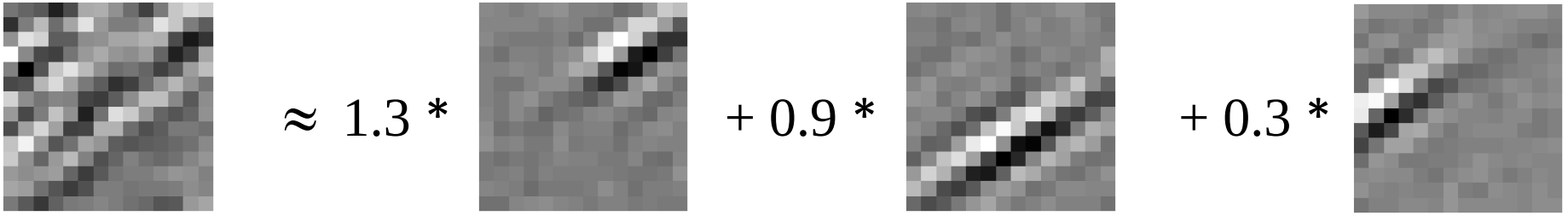
[0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, …]

= [a_1, …, a_{64}] (feature representation)

D'autres exemples



Representé par: [0, 0, …, 0, 0.6, 0, …, 0, 0.8, 0, …, 0, 0.4, …]

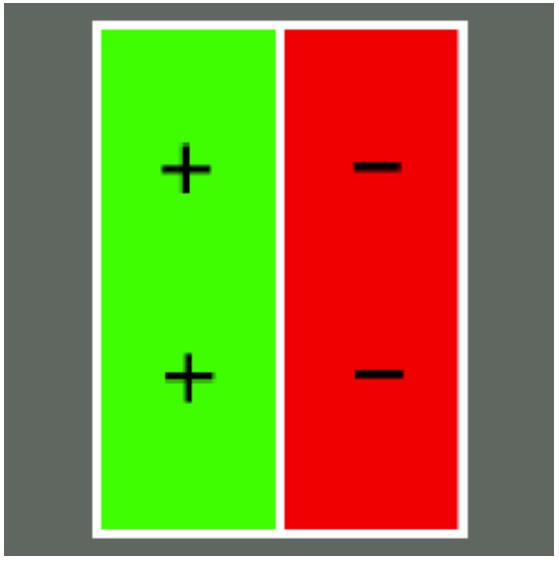


Representé par: [0, 0, …, 0, 1.3, 0, …, 0, 0.9, 0, …, 0, 0.3, …]

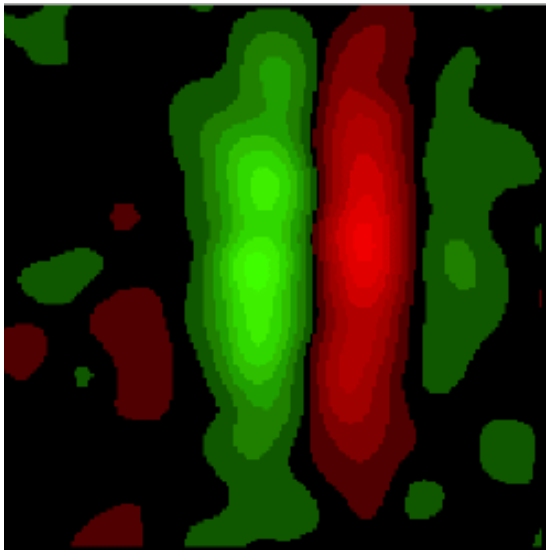
- Les images sont décomposées en patchs élémentaires, proches des filtres de Gabors
- Les coéfficients constituent une représentation plus compacte et de plus haut niveau que les pixels

Première étape du cortex visuel: V1

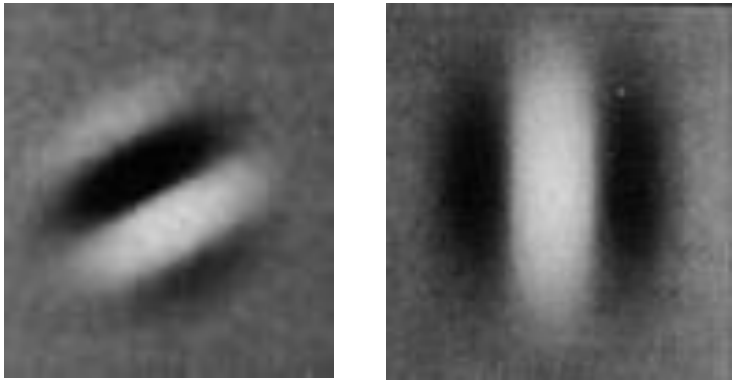
La première étape du cortex visuel fait de la “détection d'orientation” ..



Schematic of simple cell



Actual simple cell



“Gabor functions”

Sparse coding : algorithme

Entrée: patches $y^{(1)}$, $y^{(2)}$, …, $y^{(m)}$

$$\min_{\boldsymbol{\alpha}_i, \mathbf{D} \in \mathcal{C}} \sum_i \underbrace{\frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2}_{\text{reconstruction}} + \underbrace{\lambda \psi(\boldsymbol{\alpha}_i)}_{\text{sparsity}}$$

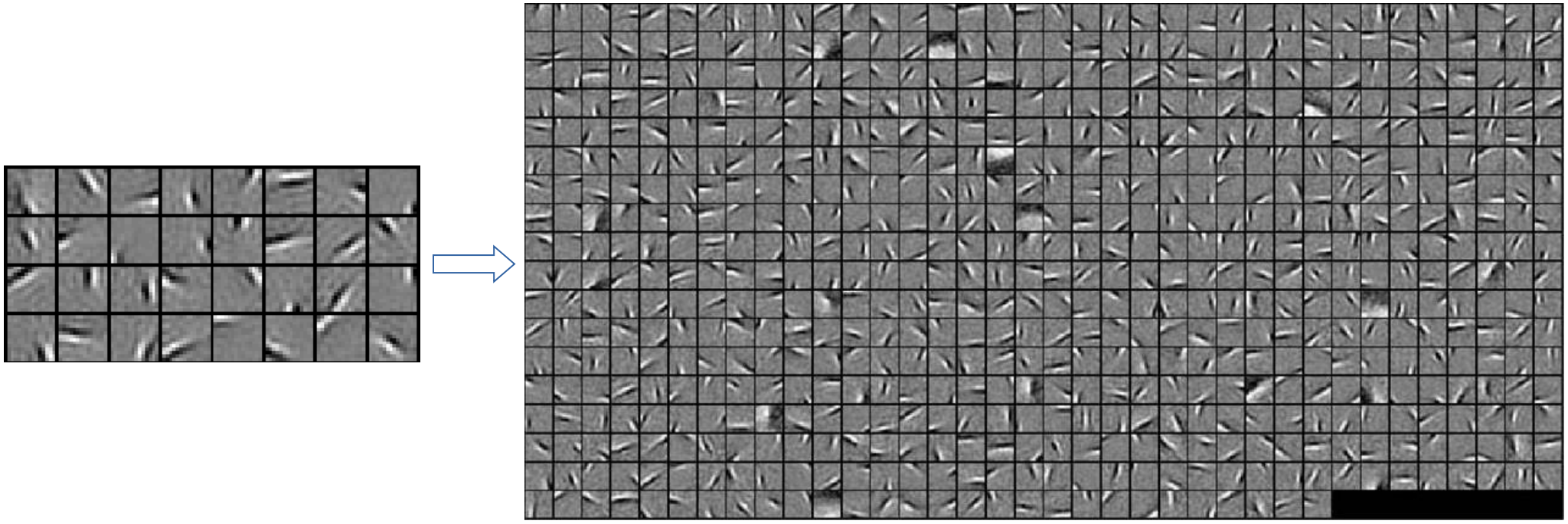
Optimisation alternée:

- fixer les alphas, minimiser sur D
- fixer D, minimiser sur les alphas

jusqu'à "convergence"

Sparse coding : algorithme

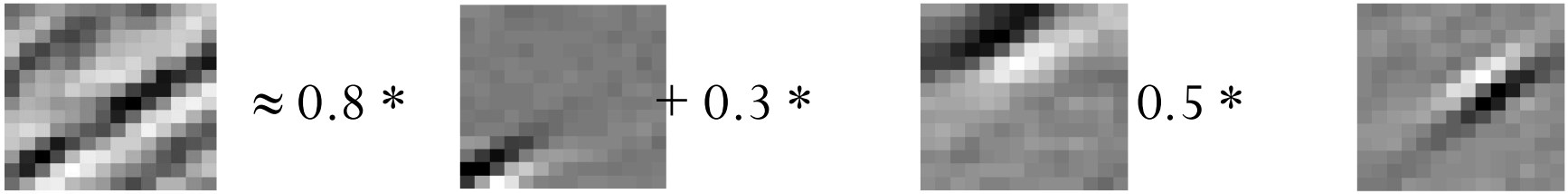
- De nombreuses améliorations pour passer à l'échelle



- "Feature-sign search algorithm"

[Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng, 2006]

Sparse coding
pour la classification d'images



[0, 0, ..., 0, **0.8**, 0, ..., 0, **0.3**, 0, ..., 0, **0.5**, ...]

- Bien meilleur qu'une représentation par pixels, mais pas aussi compétitif que les SIFT (etc)

  - Combiner tout ça avec les SIFT !!

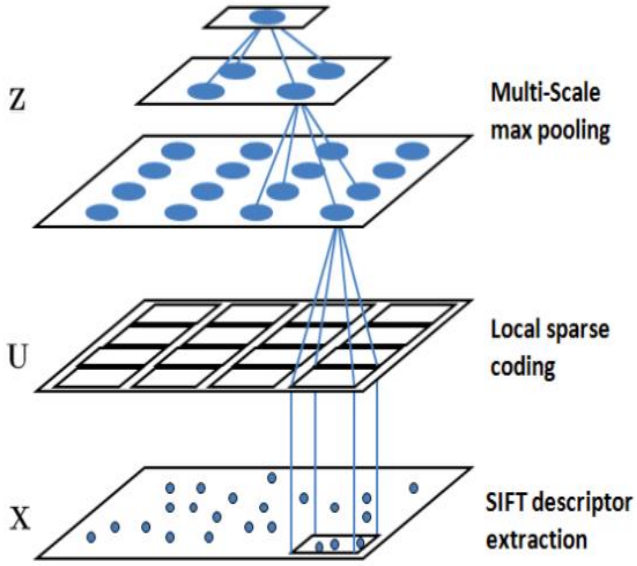
  - Deep learning ?

Sparse coding avec des SIFT

- Simplement remplacer les patchs par des vecteurs SIFT
  - apprend un dictionnaire (codebook) sparse
  - et fait le coding (*soft assignment*) en même temps

- Intègre facilement le pooling spatial

  [Yang et al., 2009]

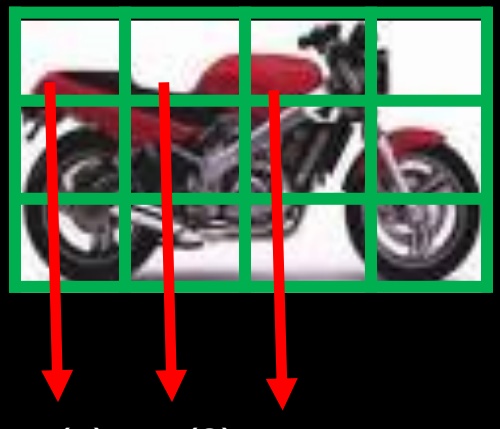


Putting it together

Feature representation

Learning algorithm

Suppose you've already learned bases $\phi_1, \phi_2, \ldots, \phi_k$. Here's how you represent an image.



$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ …

$a^{(1)}$ $a^{(2)}$ $a^{(3)}$ …

$$\begin{bmatrix} a^{(1)} \\ a^{(2)} \\ \vdots \\ a^{(12)} \end{bmatrix}$$

or $\sum_{i=1}^{12} a^{(i)}$

Learning algorithm

e.g., 73-75% on Caltech 101 (Yang et al., 2009, Boreau et al., 2009)

Du Sparse Coding au CNN

- SC produit des filtres redontants qui sont souvent des versions translatées des autres
  - Chaque patch est traité indépendamment
  - Ignore le traitement par convolution
- Idée
  - Sparse coding sur l'image entière et contraindre le dictionnaire pour grouper les filtres

$$\ell(x, z, D) = \frac{1}{2}||x - \sum_{k=1}^{K} D_k * z_k||_2^2 + |z|_1$$

avec D_k un filtre 2D de taille s x s, $\boldsymbol{x}$ une image w x h et z_k la réponse de la convolution : une image (feature map) de taille (w + s − 1) × (h + s − 1)

  - Avec non-linéarité

$$\mathcal{L}(x, z, \mathcal{D}) = \frac{1}{2}||x - \sum_{k=1}^{K} \mathcal{D}_k * z_k||_2^2 + \sum_{k=1}^{K} ||z_k - f(W^k * x)||_2^2 + |z|_1$$